UCRL-ID-144213

# Building a High Performance Raw Disk Subsystem for Alpha/Linux

*Jim E. Garlick*


**July 2, 2001**

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

# Building a High Performance Raw Disk Subsystem for Alpha/Linux

Jim E. Garlick
garlick@llnl.gov

July 2, 2001

**Abstract**

The Linux kernel version 2.2.19 lacks UNIX-style raw disk support, and its SCSI layer is optimized for small transfer sizes. This report describes kernel patches to add raw disk support and enhance performance in the SCSI layer and QLA2x00 Fibre Channel device driver for large transfer sizes. Benchmarks demonstrate raw disk performance of 191 megabytes/second write, 176 megabytes/second read for one megabyte random I/O on a Compaq ES40 computer system with two QLogic QLA2200F Fibre Channel host bus adapters, each connected to two Ciprico RF7010 arrays on arbitrated loop.

## 1 Introduction

Lawrence Livermore National Laboratory has been involved in the porting and tuning of the Frangipani(Thekkath et al., 1997) network filesystem and Petal(Lee and Thekkath, 1996) virtual disk server for a parallel scientific workload on Alpha/Linux massively parallel processors (MPP's) since early 2000. Petal's job is to provide network access to a virtual disk which may be served by multiple cluster nodes, each serving data from multiple physical disks. Part of the tuning work was to modify Petal's RPC layer to directly use the Quadrics Elan3 interconnect. This made it possible for the RPC layer to deliver nearly 200 megabytes/second for one megabyte transfers, an improvement over the 35 megabytes/second obtained with 64 kilobyte transfers using User Datagram Protocol (UDP) over the same Elan3 interconnect.

Petal runs in user space and therefore requires direct access to disk devices. Ideally, this access would be provided by a raw disk subsystem which bypasses the buffer cache; however, Linux is unique among UNIX-like operating systems in that it does not support UNIX-style raw disk access. Section 2 describes patches to the kernel which add raw device support.

Livermore's scientific workload demands transfers of large blocks from a parallel filesystem, and Frangipani's read-ahead and write-behind algorithms aggregate smaller requests when possible, resulting in a Petal access pattern that favors large blocksizes. Petal's striping across multiple disks and nodes, its mapping of virtual offsets to physical block numbers, and the fact that multiple I/O streams are served concurrently conspire to create a request pattern that is not sequential on the disk. The disk subsystem used by a Petal server should therefore be optimized for random I/O of large blocksizes. Tuning of the hardware used in this report to maximize performance for large transfers is described in Section 3.

Finally, a cost-effective Petal server should balance the performance of its interconnect with that of its disk subsystem. Since a Petal server in theory could service 200 one megabyte requests per second over the Quadrics interconnect, the raw disk subsystem on a Petal server should have comparable performance for the same workload. Section 4 demonstrates with benchmarks that this is achieved for the hardware described in this report.

## 2 UNIX-style Raw Disk Devices

Linus Torvalds, the Linux kernel's primary architect, omitted support for UNIX-style raw (unbuffered) disk access from the kernel through version 2.2.19 as a conscious design decision(Torvalds, ). Due to the demand from vendors of high-end relational database management systems and others for raw devices, Stephen Tweedie of RedHat, Inc. developed a patch[1] to implement a variant of raw device support in the Linux 2.2.X series. The Stephen Tweedie *rawio* patch has been distributed as part of RedHat Linux since version 6.1, and has been incorporated into the mainstream 2.4 kernel series.

*rawio* has two unique characteristics. First, it employs *zero-copy I/O*. Instead of copying user buffers to kernel space before initiating a *direct memory access* (DMA) to perform the I/O, the kernel sets up the DMA to operate directly on the user buffers, saving the overhead of copying data between user and kernel space. The details of preventing the system from swapping out user buffers while a DMA is pending are managed by the kernel, but zero-copy I/O does introduce one constraint in user space: buffers must be aligned on the device sector size boundary, typically 512 bytes. It is always safe to use the page-aligned buffer returned by *valloc*. It follows that the *dd* command must be modified to use an aligned buffer if it is to be used on raw devices.

The second characteristic of *rawio* is that raw device special files differ from traditional UNIX, where each block device has a corresponding character device for unbuffered I/O. Instead, *rawio* implements a set of *unbound* raw devices, `/dev/rawN`, and a control device `/dev/rawctl` used to bind them to block devices. A utility called *raw* is a front end for the `/dev/rawctl` ioctl.

*rawio* suffers from one major deficiency in our application: it makes use of the file system `buffer_head` data structure and associated queueing routines, necessitating the fragmention of large raw requests into separate one kilobyte transfers. As stated in the introduction, a goal of our work is to optimize for large transfers. This issue is addressed for SCSI devices by a patch developed at SGI[2]. The SGI patch bypasses the `buffer_head` routines and increases the maximum atomic transfer size to one megabyte.

As a bonus, the SGI patch also provides traditional UNIX character/block device special files, where character special raw SCSI devices have the same major and minor numbers as the corresponding block SCSI devices, and the same name except an "r" is prepended; for example, block device `/dev/sd1a` would correspond to raw device `/dev/rsd1a`.

The net effect of the *rawio* and SGI patches applied together to the Linux kernel version 2.2.19 is an implementation of UNIX-style raw devices with the following caveats:

- Only SCSI devices are supported (this includes Fibre Channel which uses the SCSI-FCP protocol). Other block devices such as those used to access IDE disks or meta devices like *loopback* or the *multiple disk* (MD) driver are not supported.
- Buffers must be aligned on the device sector size. A *read* or *write* request operating on an unaligned buffer will fail and set *errno* to EINVAL.
- A maximum of one megabyte can be transferred atomically. A *read* or *write* request for more than one megabyte will fail and set *errno* to EINVAL.

Many terabytes have been pushed through the raw device path on Alpha/Linux in the course of developing and testing Petal code. The implementation is stable, and its performance is demonstrated in Section 4.

## 3 Fibre Channel Disk Subsystem

The test hardware used in this report is depicted in Figure 1. It consists of a computer system, a Fibre Channel disk subsystem, and an interface to the Quadrics Elan3 interconnect. The computer

[1] ftp://ftp.linux.org.uk/pub/linux/sct/fs/raw-io/
[2] http://oss.sgi.com/projects/rawio/

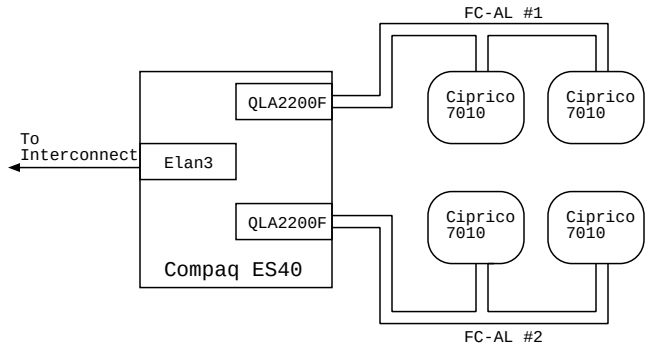


Figure 1: Hardware Test Environment

system is a Compaq ES40 configured with two gigabytes of RAM (all four memory banks populated) and four 500MHz Alpha EV6 CPU's. The ES40 has dual 64 bit, 33 MHz PCI busses; the Elan3 adapter board is on one bus, and the QLogic QLA2200F Fibre Channel *host bus adapters* (HBAs) are on the other.

Each HBA shares a Fibre Channel arbitrated loop with two Ciprico RF7010 RAID arrays, a RAID-3 array built from nine 10,000 RPM, 18 gigabyte SCSI disks (eight data disks and one parity disk). The array's capacity is 144 gigabytes, and the stripe size is four kilobytes. Its configuration, set via the front panel, is detailed in Appendix B. The remainder of this section focuses on the tuning of the QLogic HBA's.

The QLogic QLA2200F is a 64 bit, 33/66 MHz PCI adapter that supports the SCSI-3 Fibre Channel Protocol (SCSI-FCP) standard over multi-mode fiber optic media. It can transfer data at up to 100 megabytes/second.[3] Both the HBA firmware and the Linux device driver require tuning for our environment.

To change QLA2200F firmware settings when the host is an Alpha architecture system running the SRM BIOS, the HBA must be removed from the system and placed in a PC, where it will function at reduced performance in a 32 bit slot (acceptable for configuration purposes). The *Fast!UTIL* configuration utility on the PC is accessed by pressing ALT-Q during the QLA2200 BIOS initialization. The HBA manual(QLogic, 2000) describes the parameters that may be tuned via *Fast!UTIL*. For this report, factory defaults were set, then the *Frame Size* parameter in the *Host Adapter Settings* menu was increased from 1024 to 2048. The final firmware values are presented in Appendix A below.

The Linux driver for the QLA2x00 series is available from QLogic's web site[4]. We started with version 4.24-Beta. As distributed, 4.24-Beta functions on Alpha/Linux, an improvement over previous versions, but the following changes were still necessary:

- Increased SG_SEGMENTS in qla2x00.h from 32 to 144. This number is passed to the SCSI layer to inform it of the adapter's maximum scatter-gather table size. This increase is necessary to achieve good performance with large blocksizes.
- Reduced delays when reading NVRAM to avoid "spinlock stuck" messages from the kernel during initialization and module unload.
- Added code to retry failed firmware reset until it succedes. This fixes a bug where occasionally the Fibre Channel loop does not come up when the module is initialized, resulting in missing SCSI devices.

With the combination of firmware settings and driver modifications described above, the QLogic QLA2200F HBA functions quite well under Alpha/Linux 2.2.19 and in combination with the raw

[3]Fibre Channel FC-0 serial link speed is 1.0625 gigabaud, and FC-1 8B/10B encoding scheme uses 10 bits for each byte, yielding a 100 megabytes/second effective rate; this does not take into account the framing overhead of FC-2 and protocol overheads of FC-3 and FC-4.(Benner, 1996)

[4]http://www.qlogic.com/bbs-html/ts_page.html

device patches described in Section 2, two HBA's can operate at 95 or more percent of their combined maximum data transfer rate of 200 megabytes/second.

# 4 Performance Results

Three benchmarks measured data rates for the test system: *devtest* (Version 1.0), which measures random I/O; *donnie*, which measures sequential I/O across several sections of disk concurrently; and *xdd* (Version 5.3-alpha1), which measures seqential I/O.

*devtest* measures random I/O performance. For this report, *devtest* started four threads per device, meaning the *queue depth*, or number of simultaneous outstanding requests, was four per device. Requests were randomized over a 100 gigabyte section of the array. Figure 2 summarizes the results. The one megabyte write rate was measured at 191 megabytes/second and read rate at 176 megabytes/second.

*donnie* is a derivative of the *bonnie*[5] benchmark that operates on raw devices. The High Performance Storage System (HPSS) group at Livermore uses it to evaluate storage subsystems. *donnie* performs I/O sequentially to a number of *files* (actually contiguous segments of the target device) of various sizes. A separate concurrent thread executes for each file For this report, *donnie* performed I/O on four files on each of four arrays, thus the queue depth per array was four. Figure 3 shows output of the *donnie* benchmark. Read and write rates for one megabyte transfers were measured at 168 megabytes/second.

*xdd*(Ruwart and O'Keefe, 1995) is a raw I/O benchmark developed at the University of Minnesota. In our tests, the queue depth was set to one for each device. A report(Ruwart and Elder, 2000) prepared for Livermore uses *xdd* to measure raw performance of a Ciprico/QLogic Fibre Channel subsystem similar[6] to ours, but hosted on an SGI ONYX running IRIX. We hoped to reproduce the report's results up to the one megabyte atomic transfer size limit imposed by Linux raw devices.

Figure 4 depicts *xdd* read and write test results. Reads peaked at 188 megabytes/second; writes at 162 megabytes/second. There was a suprise when the SGI system results were compared with the same configuration on Linux. For one megabyte transfers using two arrays and two adapters, the SGI system reported read and write rates of approximately 175 megabyte/second, while Linux reported a read rate of 173 megabytes/second and a write rate of 142 megabytes/second.

To determine if *xdd* accurately reports write rates on Linux, a version of *devtest*, modified to perform sequential I/O in the same manner as *xdd*, measured sequential performance. Figure 5 shows the results. For one megabyte transfers on four arrays and two adapters, *devtest* reported a sequential read rate of 183 megabytes/second and a write rate of 188 megabyte/second. The rates for one megabyte transfers on two arrays and two adapters of 167 megabytes/second read and write compare favorably to the SGI system results quoted above, in terms of both overall performance, and the similarity between read and write rates.

In summary, the important performance results are *devtest* rates for random one megabyte transfers, since this pattern of access most closely matches that anticipated for a Petal server in our environment. The *devtest* 191 megabytes/second write and 176 megabytes/second read rates come close to the rates possible over Petal RPC; therefore, a Petal server configured as described in this report meets the goal of balancing performance of the raw disk subsystem with that of the interconnect.

# 5 Conclusion

Obtaining support in the Linux kernel version 2.2.19 for raw devices and high performance on the hardware described in this report consists of applying two patches to the kernel source code, modifying the Linux device driver for the QLogic HBA, and setting up the firmware of the HBA and the disk arrays.

[5] http://www.textuality.com/bonnie/

[6] The SGI system was configured with Ciprico RF7000 arrays populated with Seagate Barricuda 50 gigabyte drives (7200 RPM?), compared to our 18 gigabyte IBM drives (10,000 RPM); and QLogic host adapters (model unknown) plugged into PCI to XIO adapters.

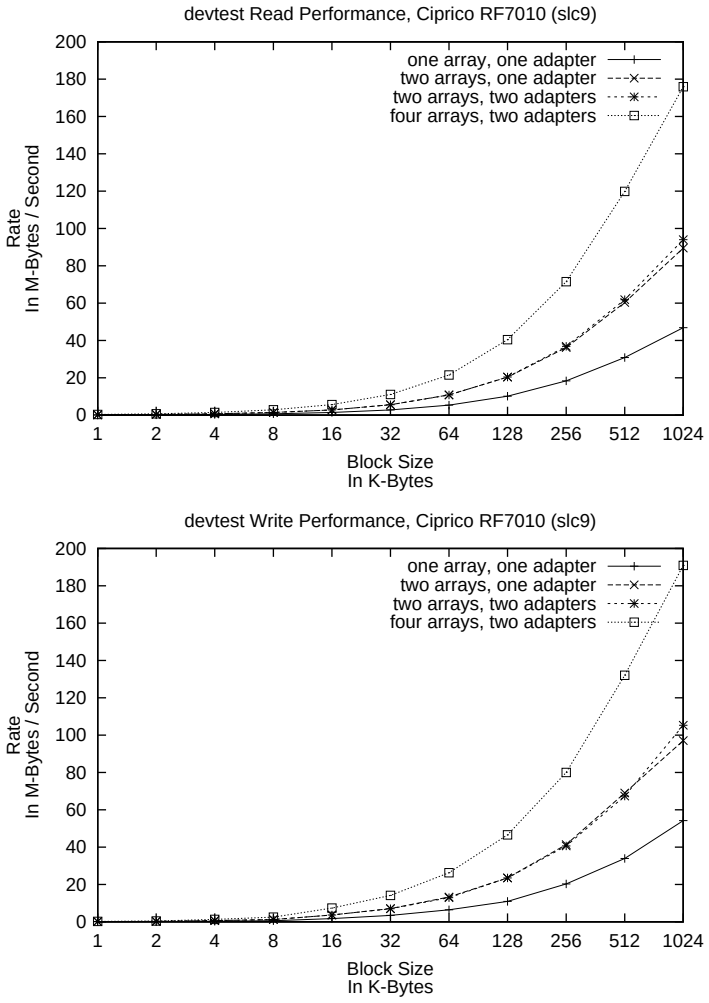


Figure 2: *devtest* Read and Write Performance - Ciprico RF7010

```
RUN BEGINNING Thu May 24 15:23:05 2001

Linux slc9 2.2.19raw_smp #6 SMP Tue May 15 14:54:10 PDT 2001 alpha unknown

         4 devices
         1 partitions per device
    131072 megabytes per device
    131072 megabytes per partition
         4 simultaneous jobs (file sizes) per partition
      1024 kilobytes per buffer
       300 seconds job duration (for each - read and write)

/dev/rsdb /dev/rsdc /dev/rsdd /dev/rsde

WRITE DATA - 300.9 SECONDS RUN TIME

device 0 data
mbyte/file  bufs wrtn   MB written  MB per sec   Utime   Stime   Clock  %cpu
     65536       3262     3262.000      10.843    0.01    0.61  300.84   0.2
     16384       2980     2980.000       9.906    0.01    0.54  300.83   0.2
      4096       3145     3145.000      10.456    0.01    0.59  300.79   0.2
      1024       2902     2902.000       9.646    0.00    0.52  300.85   0.2
Write Total     12289    12289.000      40.843    0.03    2.25           0.8

device 1 data
mbyte/file  bufs wrtn   MB written  MB per sec   Utime   Stime   Clock  %cpu
     65536       3120     3120.000      10.373    0.00    0.57  300.79   0.2
     16384       3016     3016.000      10.028    0.00    0.58  300.75   0.2
      4096       3020     3020.000      10.040    0.01    0.58  300.81   0.2
      1024       3376     3376.000      11.225    0.00    0.60  300.76   0.2
Write Total     12532    12532.000      41.657    0.02    2.33           0.8

device 2 data
mbyte/file  bufs wrtn   MB written  MB per sec   Utime   Stime   Clock  %cpu
     65536       3316     3316.000      11.023    0.00    0.58  300.81   0.2
     16384       2968     2968.000       9.867    0.00    0.52  300.79   0.2
      4096       3352     3352.000      11.144    0.00    0.60  300.78   0.2
      1024       3243     3243.000      10.784    0.01    0.61  300.73   0.2
Write Total     12879    12879.000      42.811    0.03    2.32           0.8

device 3 data
mbyte/file  bufs wrtn   MB written  MB per sec   Utime   Stime   Clock  %cpu
     65536       3301     3301.000      10.976    0.01    0.60  300.75   0.2
     16384       3088     3088.000      10.264    0.00    0.59  300.85   0.2
      4096       3193     3193.000      10.617    0.00    0.55  300.74   0.2
      1024       3380     3380.000      11.238    0.00    0.61  300.77   0.2
Write Total     12962    12962.000      43.085    0.02    2.35           0.8

Grand Total     50662    50662.000     168.371    0.09    9.25           3.1

READ DATA - 301.0 SECONDS RUN TIME

device 0 data
mbyte/file  bufs read      MB read  MB per sec   Utime   Stime   Clock  %cpu
     65536       2996     2996.000       9.959    0.01    0.59  300.84   0.2
     16384       3312     3312.000      11.007    0.01    0.58  300.89   0.2
      4096       2984     2984.000       9.917    0.00    0.53  300.89   0.2
      1024       2856     2856.000       9.494    0.00    0.53  300.82   0.2
Read Total      12148    12148.000      40.370    0.02    2.23           0.7

device 1 data
mbyte/file  bufs read      MB read  MB per sec   Utime   Stime   Clock  %cpu
     65536       3082     3082.000      10.241    0.00    0.57  300.94   0.2
     16384       3244     3244.000      10.782    0.01    0.58  300.88   0.2
      4096       3169     3169.000      10.531    0.01    0.60  300.92   0.2
      1024       2955     2955.000       9.823    0.01    0.53  300.81   0.2
Read Total      12450    12450.000      41.367    0.02    2.28           0.8

device 2 data
mbyte/file  bufs read      MB read  MB per sec   Utime   Stime   Clock  %cpu
     65536       3136     3136.000      10.423    0.01    0.57  300.86   0.2
     16384       3497     3497.000      11.626    0.01    0.60  300.79   0.2
      4096       2962     2962.000       9.846    0.01    0.56  300.83   0.2
      1024       3387     3387.000      11.258    0.01    0.59  300.86   0.2
Read Total      12982    12982.000      43.147    0.03    2.32           0.8

device 3 data
mbyte/file  bufs read      MB read  MB per sec   Utime   Stime   Clock  %cpu
     65536       3575     3575.000      11.880    0.00    0.66  300.92   0.2
     16384       3521     3521.000      11.699    0.01    0.64  300.96   0.2
      4096       3211     3211.000      10.674    0.01    0.61  300.82   0.2
      1024       2804     2804.000       9.320    0.00    0.58  300.86   0.2
Read Total      13111    13111.000      43.564    0.02    2.49           0.8

Grand Total     50691    50691.000     168.427    0.10    9.32           3.1

Average Score          101353.000     168.399                           3.1

RUN FINISHED Thu May 24 15:33:07 2001
```

Figure 3: *donnie* Results - Ciprico RF7010

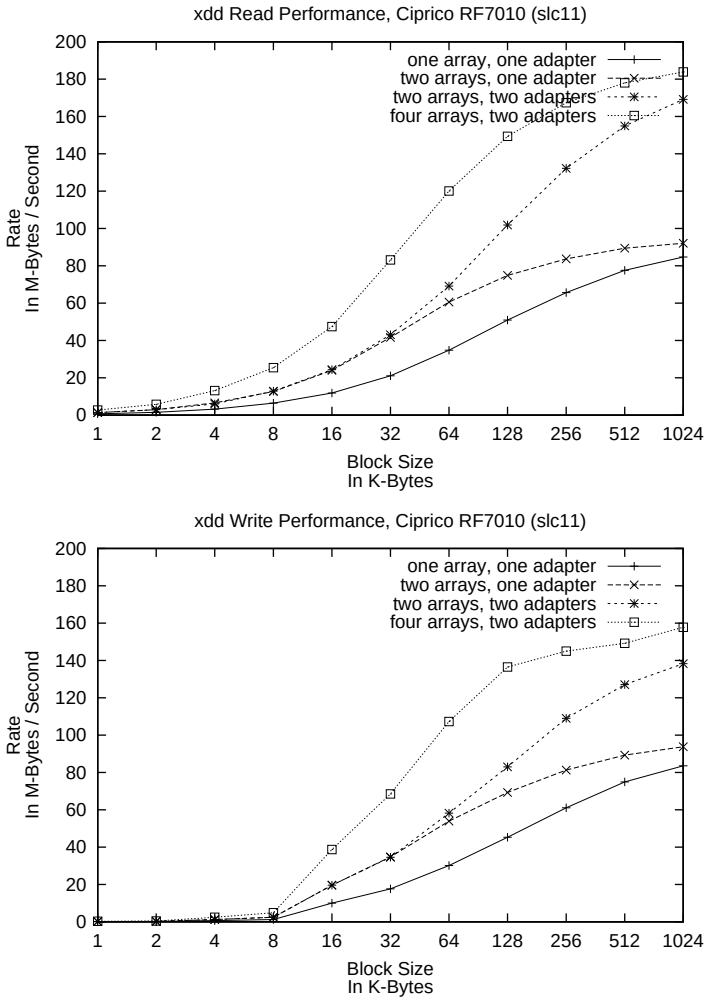


Figure 4: *xdd* Read and Write Performance - Ciprico RF7010

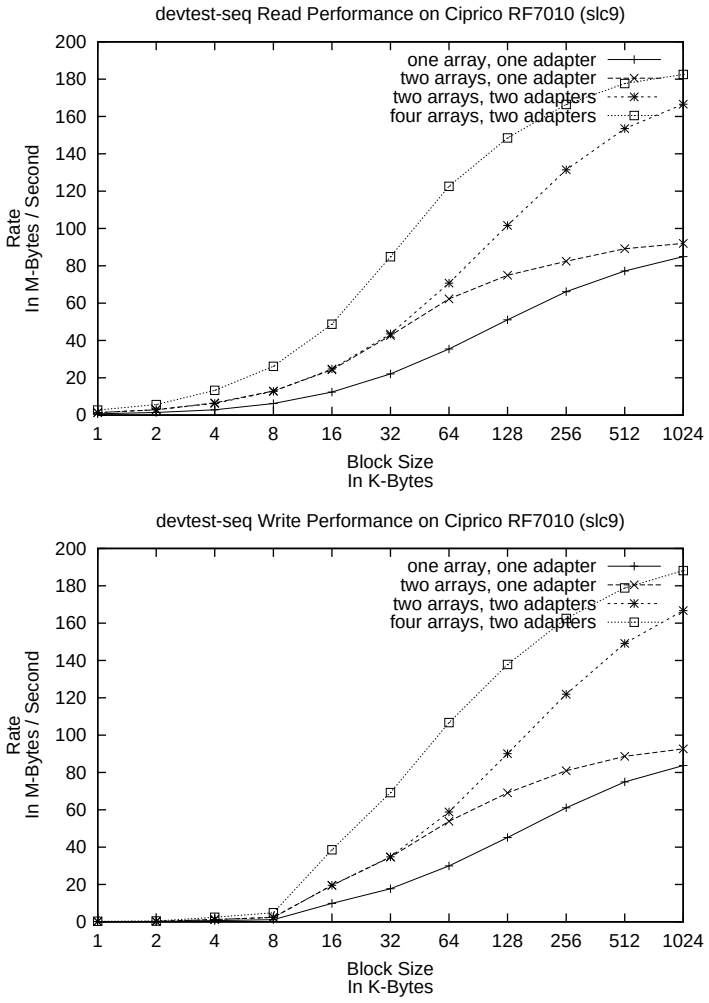


Figure 5: *devtest-seq* Read and Write Performance - Ciprico RF7010

I/O rates that are between 87 and 95 percent of the theoretical HBA bandwidth of 200 megabytes/second for random one megabyte transfers were demonstrated, meeting the goal stated in Section 1 of balancing the performance of the disk subsystem with that of the Petal RPC layer, which can transfer one megabyte blocks at a rate approaching 200 megabytes/second over the Quadrics Elan3 interconnect.

# 6 Acknowledgements

Brian Pomerantz did most of raw device work described in this report, modulo a few bug fixes, a kernel revision, and some QLogic and Ciprico firmware changes; Reto Baettig wrote the devtest program; Andrew Uselton added functionality to the devtest program and assisted with QLogic firmware and Ciprico configuration; Marcus Miller fixed bugs in the QLA2X00 Linux driver; and Danny Auble assisted with the QLogic firmware configuration and adapter installations.

# A QLogic QLA2200F Firmware Settings

The following table summarizes the QLogic QLA2200F tunable firmware settings used in this report. The hardware manual(QLogic, 2000) describes these settings in detail.

| *Adapter Settings* | |
|---|---|
| BIOS Rev | 1.54 |
| Frame Size | 2048 |
| Loop Reset Delay | 5 |
| Adapter Hard Loop ID | Disabled |
| *Advanced Adapter Settings* | |
| Execution Throttle | 16 |
| Fast Command Posting | Enabled |
| >4Gbyte Addressing | Disabled |
| Luns per Target | 8 |
| Enable LIP Reset | No |
| Enable LIP Full Login | Yes |
| Enable Target Reset | Yes |
| Login Retry Count | 8 |
| Part Down Retry Count | 8 |
| Drivers Load RISC Code | Enabled |
| Enable Database Updates | No |
| Disable Database Load | No |
| IOCB Allocation | 256 |
| Extended Error Logging | Disabled |
| *Extended Settings* | |
| Ext control block | 0 |
| RIO op mode | 3 |
| connection op | Disabled |
| class 2 svc | Disabled |
| ack 0 | Disabled |
| fc tape | Disabled |
| fc confirm | Disabled |
| cmd reset num | Disabled |
| read xfer rdy | Disabled |
| reop timer | 0 |
| int delay timer | 0 |

# B Ciprico RF7010 Firmware Settings

The following table summarizes the Ciprico RF7010 configurable array options used in this report. The array service guide(Ciprico, 2000) and RAID controller manual(Ciprico, 1998) provide detailed information about configuration and array specifications.

| *Array Options* | |
|---|---|
| AL_PA | E4 |
| AL_SELID | 02 |
| ALTERNATE WWN | 000000 |
| UNIT ATTENTION | ON |
| WRITE CACHE | ON |
| AUTOSTART | ON |
| ALARM | ON |
| USE FIRMWARE | FACTORY FW |
| SPINUP TIME | 1.0 SEC |
| FC TOPOLOGY | AUTO NO FAB |
| NUM INITIATORS | 10 |

# C Linux Kernel Configuration

The `.config` file used to build the Linux kernel (version 2.2.19) used in this report is shown below. Of particular note are `CONFIG_RAW`, `CONFIG_SCSI_MULTI_LUN`, and `CONFIG_SCSI_QLOGIC_2x00`.

```
CONFIG_EXPERIMENTAL=y

CONFIG_MODULES=y
CONFIG_KMOD=y

CONFIG_ALPHA_DP264=y
CONFIG_PCI=y
CONFIG_ALPHA_EV6=y
CONFIG_ALPHA_TSUNAMI=y
CONFIG_ALPHA_SRM=y
CONFIG_SMP=y
CONFIG_PCI_OLD_PROC=y
CONFIG_NET=y
CONFIG_SYSVIPC=y
CONFIG_SYSCTL=y
CONFIG_BINFMT_AOUT=y
CONFIG_BINFMT_ELF=y
CONFIG_BINFMT_MISC=y
CONFIG_BINFMT_EM86=y
CONFIG_PARPORT=m
CONFIG_PARPORT_PC=m

CONFIG_BLK_DEV_FD=y
CONFIG_BLK_DEV_IDE=y

CONFIG_BLK_DEV_IDECD=y
CONFIG_BLK_DEV_IDESCSI=m
CONFIG_BLK_DEV_IDEPCI=y
CONFIG_BLK_DEV_IDEDMA=y
CONFIG_IDEDMA_AUTO=y

CONFIG_BLK_DEV_LOOP=m
CONFIG_BLK_DEV_NBD=y
CONFIG_BLK_DEV_RAM=y
CONFIG_BLK_DEV_RAM_SIZE=4096
CONFIG_BLK_DEV_INITRD=y
CONFIG_PARIDE_PARPORT=m

CONFIG_PACKET=y
CONFIG_FILTER=y
CONFIG_UNIX=y
CONFIG_INET=y
CONFIG_IP_MULTICAST=y
CONFIG_IP_ROUTER=y

CONFIG_SKB_LARGE=y

CONFIG_SCSI=y

CONFIG_BLK_DEV_SD=y
CONFIG_CHR_DEV_ST=m
CONFIG_BLK_DEV_SR=y
CONFIG_BLK_DEV_SR_VENDOR=y
CONFIG_CHR_DEV_SG=m

CONFIG_SCSI_MULTI_LUN=y
CONFIG_RAW=y
CONFIG_SCSI_CONSTANTS=y

CONFIG_SCSI_AIC7XXX=y
CONFIG_AIC7XXX_TCQ_ON_BY_DEFAULT=y
CONFIG_AIC7XXX_CMDS_PER_DEVICE=8
CONFIG_SCSI_SYM53C8XX=y
CONFIG_SCSI_NCR53C8XX_DEFAULT_TAGS=8
CONFIG_SCSI_NCR53C8XX_MAX_TAGS=32
CONFIG_SCSI_NCR53C8XX_SYNC=40
CONFIG_SCSI_NCR53C8XX_PQS_PDS=y
CONFIG_SCSI_QLOGIC_ISP=m
CONFIG_SCSI_QLOGIC_2x00=m

CONFIG_NETDEVICES=y

CONFIG_DUMMY=m

CONFIG_NET_ETHERNET=y
CONFIG_NET_EISA=y
CONFIG_DE4X5=m
CONFIG_DEC_ELCP=m
CONFIG_EEXPRESS_PRO100=m

CONFIG_ACENIC=m

CONFIG_VT=y
CONFIG_VT_CONSOLE=y
CONFIG_SERIAL=y
CONFIG_SERIAL_CONSOLE=y
CONFIG_UNIX98_PTYS=y
CONFIG_UNIX98_PTY_COUNT=256
CONFIG_PRINTER=m
CONFIG_PRINTER_READBACK=y
CONFIG_MOUSE=y

CONFIG_PSMOUSE=y

CONFIG_FAT_FS=m
CONFIG_MSDOS_FS=m
CONFIG_VFAT_FS=m
CONFIG_ISO9660_FS=y
CONFIG_PROC_FS=y
CONFIG_DEVPTS_FS=y
CONFIG_EXT2_FS=y

CONFIG_NFS_FS=y
CONFIG_NFS_V3=y
CONFIG_NFSD=m
CONFIG_SUNRPC=y
CONFIG_LOCKD=y

CONFIG_BSD_DISKLABEL=y
CONFIG_NLS=y

CONFIG_NLS_DEFAULT="cp437"
CONFIG_NLS_CODEPAGE_437=m
CONFIG_NLS_ISO8859_1=m
CONFIG_NLS_ISO8859_15=m

CONFIG_VGA_CONSOLE=y

CONFIG_MATHEMU=y
CONFIG_MAGIC_SYSRQ=y
```

# References


Benner, A. F.: 1996, *Fibre Channel, Gigabit Communications and I/O for Computer Networks*, McGraw-Hill

Ciprico: 1998, *Ciprico 7000 Controller Board Reference Manual*, http://www.ciprico.com/

Ciprico: 2000, *Ciprico 7000 User and Service Guide*, http://www.ciprico.com/

Lee, E. K. and Thekkath, C. A.: 1996, in *Proceedings of the Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*, pp 84–92, Cambridge, MA

QLogic: 2000, *Hardware Installation Guide for the QLA2200/2200F/2202F/2200G/2200L Fiber Channel Host Adapter for the PCI Bus*, http://www.qlogic.com/

Ruwart, T. and Elder, A.: 2000, *SAN/CXFS Test Report to LLNL*, Technical report, University of Minnesota, Laboratory for Computational Science and Engineering

Ruwart, T. M. and O'Keefe, M. T.: 1995, in *Proceedings of the Fourth NASA Goddard Conference on Mass Storage Systems and Technologies*, College Park, MD

Thekkath, C. A., Mann, T., and Lee, E. K.: 1997, in *Symposium on Operating Systems Principles*, pp 224–237

Torvalds, L., *Email from Linus Torvalds: Re: PATCH: Raw device IO for 2.1.131*, http://lwn.net/1998/1217/a/dio-lt.html

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551